Ejercicio de usabilidad de Ghosts

Ricardo Jacas

4 de diciembre de 2013

Problema 1, Tabla de Puntajes Bowling

Se desea hacer un programa que mantenga el puntaje de un jugador en un juego de *bowling*.

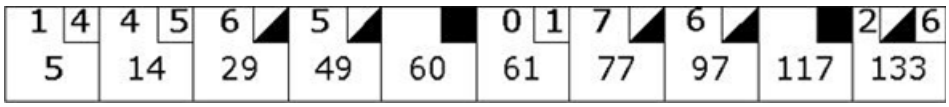

Figura 1: Puntaje de un juego de *Bowling*.

Para los efectos prácticos, se creará una clase *BowlingScore* que mantendrá el puntaje de un jugador.

El puntaje para cada cuadrilla, ver Figura 1, deberá ser obtenido mediante un método `getScore(int a)` donde `a` será la cuadrilla de la cual se está preguntando el puntaje. Además se deberá proveer un método `getTotalScore()` que retorne el puntaje total, en el momento, y un método `throwBall(int score)` para indicar cada tiro del jugador. A continuación se presenta un ejemplo de cómo se vería un test usando dichas funciones.

```
public void testSimple() {
        BowlingScore bs = new BowlingScore();
        bs.throwBall(3);
        bs.throwBall(4);
        bs.throwBall(5);
        assertEquals(bs.getTotalScore(), 12);
        bs.throwBall(3);
        assertEquals(bs.getScore(1), 7);
        assertEquals(bs.getScore(2), 15);
        assertEquals(bs.getTotalScore(), 15);
}
```

Tenga presente que:

- Cuando se realiza un *Spare* (Se botan todos los pinos en el segundo tiro de una cuadrilla), el puntaje asignado a dicha cuadrilla es `PuntajeActual + 10 + PuntajeSiguienteTiro`.
- Cuando se realiza un *Strike* (Se botan todos los pinos en el primer tiro de una cuadrilla), el puntaje asignado a dicha cuadrilla es
  `PuntajeActual + 10 + PuntajeSiguienteTiro + PuntajeSubSiguienteTiro`.
- En la décima cuadrilla se pueden realizar 3 tiros (se puede, potencialmente, obtener 30 puntos en sólo esa cuadrilla).

Antes de comenzar a completar su clase *BowlingPlayer*, escriba los casos de prueba que debe cumplir según los requerimientos indicados. Para ello, cree una clase *TestBowlingPlayer* que extienda de *TestCase* y escriba los siguientes *test*:

- *TestSimpleScore*: Para probar si al lanzar tiros sin *spare* o *strike* el puntaje en cada cuadrilla se despliega correctamente, no tiene por qué probar las 10 cuadrillas.
- *TestSpareScore*: Para probar si al lanzar tiros con *spare* se calcula correctamente el puntaje, nuevamente, no tiene para que probar las 10 cuadrillas. Aquí es fundamental probar si *getTotalScore* es consistente.
- *TestStrikeScore*: Para probar si al lanzar tiros con *strike* el puntaje se calcula correctamente. Nuevamente, ponga especial cuidado en lo que retorna *getTotalScore*.

- *TestFullGame*: Para probar el comportamiento de un juego completo, ojala con los 3 tipos de tiros. Ponga atención a la forma en que funciona la décima cuadrilla.

  Tenga en mente que para inicializar sus variables, sólo una vez, puede utilizar el método `setUp()` y declararlas en la clase, no en cada método *test*.

Pregunta 2, Números Romanos

Los números romanos son un sistema número basado en la combinación de una serie de letras del alfabeto latino. Los números pueden ser representados según las reglas que se indican a continuación.

Símbolo	I	V	X	L	C	D	M
Valor	1	5	10	50	100	500	1000

- El numeral I puede ser puesto delante de V y X para denotar 4 (IV) y 9 (IX)
- X puede ser puesto delante de L y C para denotar 40 (XL) y 90 (XC)
- C puede aparecer delante de D y M para denotar 400 (CD) y 900 (CM) siguiendo el mismo patrón.

Se desea implementar un sistema que, al recibir un número en romano lo transforme en un número entero, y viceversa. Para ello se pide que cree una clase Roman que acepte dos constructores, `Roman(String a)` y `Roman(int a)` para recibir un número en romano o como entero, respectivamente. La clase debe implementar los métodos `getAsInt()` para operar el número con enteros y los métodos `add(Roman b)`, `minus(Roman b)` y `mult(Roman b)` para operar dos romanos. Ojo que estos métodos retornan romanos.

Antes de comenzar a completar su clase *Roman*, escriba los casos de prueba que debe cumplir según los requerimientos indicados. Para ello, cree una clase *TestRoman* que extienda de *TestCase* y escriba los siguientes *tests*:

- *TestNumberValue*: Para probar si los casos mostrados en la tabla se transforman correctamente.
- *TestNumberFormat*: Para probar si los casos 4, 9, 40, etc, funcionan correctamente.
- *TestBigNumber*: Para probar números que combienen los casos anteriores. Trate de probar combinaciones cotidianas de los números que sean caso especial (ejemplo, 1949).
- *TestIntConvertion*: Para probar las transformaciones a enteros.
- *TestAdd*: Para probar el sumado de romanos.
- *TestMinus*: Para probar la resta de romanos.
- *TestMult*: Para probar la multiplicación de romanos.

Tenga en mente que para inicializar sus variables, sólo una vez, puede utilizar el método `setUp()` y declararlas en la clase, no en cada método *test*.